43장. Ajax

43.1 Ajax란?

Ajax(Asynchronous JavaScript and XML)

: 자바스크립트를 사용하여 브라우저가 서버에게 비동기 방식으로 데이터를 요청하고, 서버가 응답한 데이터를 수신하여 웹페이지를 동적으로 갱식하는 프로그래밍 방식

1. Ajax는 브라우저에서 제공하는 Web API인 XMLHttpRequest 객체를 기반으로 동작

   XMLHttpRequest는 HTTP 비동기 통신을 위한 메서드와 프로퍼티를 제공

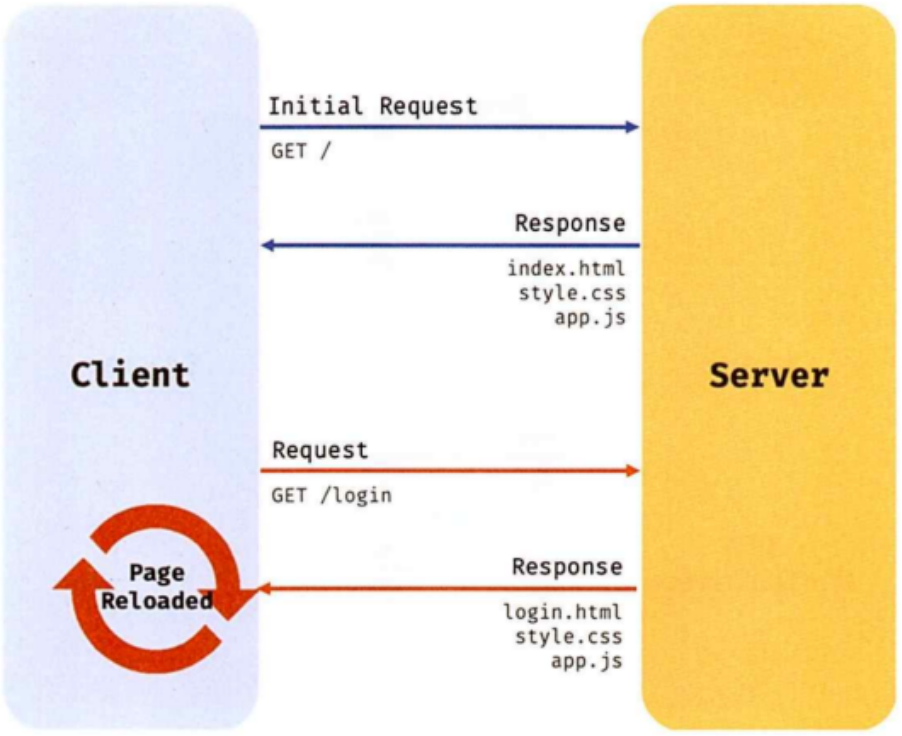
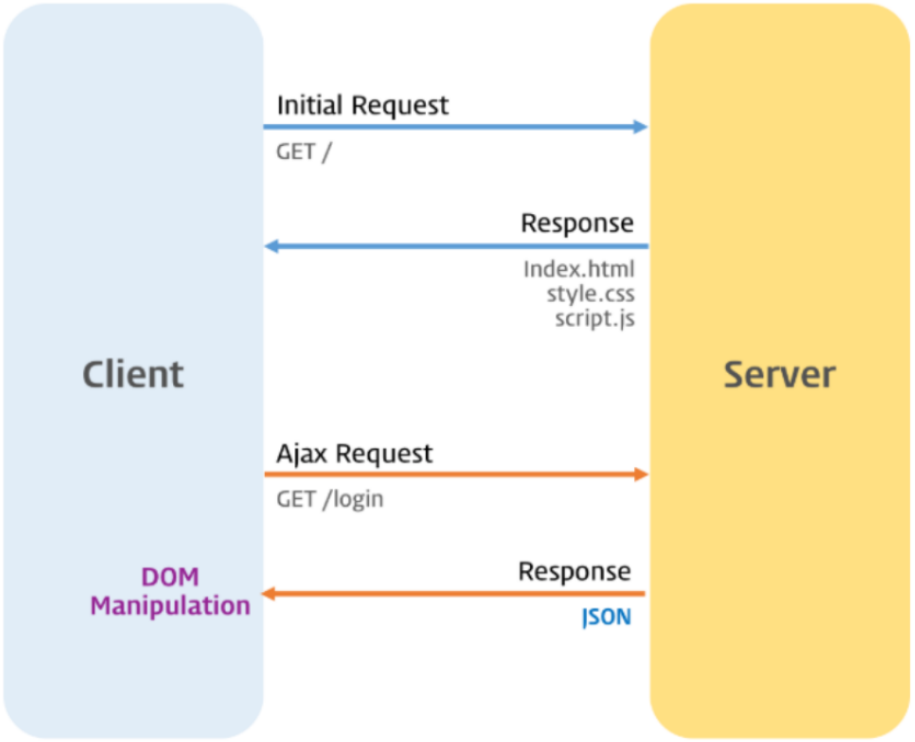


그림 43-1 전통적인 웹페이지의 생명 주기



2. 전통적 방식과 비교했을 때 Ajax의 **장점**
   - 변경 부분 갱신하는데 필요한 데이터만 서버로부터 전송 받음 → 불필요한 데이터 통신 발생 X
   - 변경 필요 없는 부분은 다시 렌더링 X → 화면이 순간적으로 깜박이는 현상 발생 X
   - 클라이언트-서버 간 통신이 비동기 방식으로 동작 → 서버에게 요청 보낸 이후 블로킹 발생 X

43.2 JSON

JSON(JavaScript Object Notation)

: 클라이언트와 서버간의 HTTP 통신을 위한 텍스트 데이터 포맷

43.2.1 JSON 표기 방식

1. JSON은 자바스크립트 객체 리터럴과 유사하게 키와 값으로 구성된 순수 텍스트.
   - 키, 문자열 : 반드시 큰따옴표("")로 묶어야 함
   - 값 : 객체 리터럴과 같은 표기법 그대로 사용 가능

43.2.2 JSON.stringfy

1. 객체를 JSON 포맷의 문자열로 변환.

2. 클라이언트가 서버로 객체를 전송하려면, 객체를 문자열화해야 함.

   = **직렬화(serializing)**

3. JSON.stringfy 메서든느 배열도 JSON 포맷의 문자열로 변환.

43.2.3 JSON.parse

1. JSON 포맷의 문자열을 객체로 변환 (= **역직렬화**)

43.3 XMLHttpRequest

- 자바스크립트를 사용하여 HTTP 요청을 전송하려면 XMLHttpRequest 객체 사

43.3.1 XMLHttpRequest 객체 생성

1. XMLHttpRequest 객체는 XMLHttpRequest 생성자 함수를 호출하여 생성

```
// XMLHttpRequest 객체의 생성
const xhr = new XMLHttpRequest();
```

43.3.2 XMLHttpRequest 객체의 프로퍼티와 메서드

XMLHttpRequest 객체의 프로토타입 프로퍼티

프로토타입 프로퍼티	설명
readyState	HTTP 요청의 현재 상태를 나타내는 정수, 다음과 같은 XMLHttpRequest의 정적 프로퍼티를 값으로 갖는다.- **UNSENT: 0**- **OPENED: 1**- **HEADERS_RECEVIED: 2**- **LOADING: 3**- **DONE: 4**
status	HTTP 요청에 대한 응답 상태(HTTP 상태 코드)를 나타내는 정수**예) 200**
statusText	HTTP 요청에 대한 응답 메세지를 나타내는 문자열**예) "OK"**
responseType	HTTP 응답 타입**예) document, json, text, blob, arraybuffer**
response	HTTP 요청에 대한 응답 몸체(reseponse body), responseType에 따라 타입이 다르다.
responseText	서버가 전송한 HTTP 요청에 대한 응답 문자열

XMLHttpRequest 객체의 이벤트 핸들러 프로퍼티

이벤트 핸들러 프로퍼티	설명
onreadystatechange	readyState 프로퍼티 값이 변경된 경우
onloadstart	HTTP 요청에 대한 응답을 받기 시작한 경우

onprogress	HTTP 요청에 대한 응답을 받는 도중 주시적으로 발생
onabort	abort 메서드에 의해 HTTP 요청이 중단된 경우
onerror	HTTP 요청에 에러가 발생한 경우
onload	HTTP 요청이 성공적으로 완료한 경우
ontimeout	HTTP 요청이 초과한 경우
onloadend	HTTP 요청이 완료한 경우. HTTP 요청이 성공 또는 실패하면 발생

XMLHttpRequest 객체의 메서드

메서드	설명
open	HTTP 요청 초기화
send	HTTP 요청 전송
abort	이미 전송된 HTTP 요청 중단
setRequestHeader	특정 HTTP 요청 헤더의 값을 설정
getRequestHeader	특정 HTTP 요청 헤더의 값을 문자열로 반환

XMLHttpRequest 객체의 정적 프로퍼티

정적 프로퍼티	값	설명
UNSENT	0	open 메서드 호출 이전
OPEND	1	open 메서드 호출 이후
HEADERS_RECEIVED	2	send 메서드 호출 이후
LOADING	3	서버 응답 중(응답 데이터 미완성 상태)
DONE	4	서버 응답 완료

43.3.3 HTTP 요청 전송

1. HTTP 요청을 전송하는 경우 다음 순서를 따른다.
    a. XMLHttpRequest.prototype.open 메서드로 HTTP 요청을 초기화
    b. 필요에 따라 XMLHttpRequest.prototype.setRequestHeader 메서드로 특정 HTTP 요청의 헤더 값을 설정
    c. XMLHttpRequest.prototype.send 메서드로 HTTP 요청을 전송

```
// XMLHttpRequest 객체 생성
const xhr = new XMLHttpRequest();

// HTTP 요청 초기화
xhr.open('GET', '/user');

// HTTP 요청 헤더 설정
// 클라이언트가 서버로 전송할 데이터의 MIME 타입 지정:json
xhr.setRequestHeader('content-type', 'application/json');
```

```
// HTTP 요청 전송
xhr.send();
```

XMLHttpRequest.prototype.open

```
xhr.open(method, url[, async])
```

매개변수	설명
method	HTTP 요청 메서드("GET", "POST", "PUT", "DELETE" 등)
url	HTTP 요청을 전송할 URL
async	비동기 요청 여부, 옵션으로 기본값 true이며, 비동기 방식으로 동작한다.

HTTP 요청 메서드는 클라이언트가 서버에게 요청의 종류와 목적(리소스에 대한 행위)를 알리는 방법이다. 주로 5가지 요청 메서드(GET, POST, PUT, PATCH, DELETE 등)을 사용하여 CRUD를 구현한다.

HTTP 요청 메서드	종류	목적	페이로드
GET	index/retrieve	모든/특정 리소스 취득	❌
POST	create	리소스 생성	⭕
PUT	replace	리소스의 전체 교체	⭕
PATCH	modify	리소스의 일부 수정	⭕
DELETE	delete	모든/특정 리소스 삭제	❌

XMLHttpRequest.prototype.send

- GET 요청 메서드의 경우 데이터를 URL의 일부분인 쿼리 문자열로 서버에 전송한다.
- POST 요청 메서드의 경우 데이터(페이로드)를 요청 몸체(request body)에 담아 전송한다.

HTTP 요청 메서드가 GET인 경우 send 메서드에 페이로드로 전달한 인수는 무시되고 요청 몸체는 null로 설정된다.

XMLHttpRequest.prototype.setRequestHeader

1. Content-type은 요청 몸체에 담아 전송할 데이터의 MIME 타입의 정보를 표현한다. 자주 사용되는 MIME 타입은 다음과 같다.

MIME 타입	서브타입
text	text/plain, text/html, text/css, text/javascript
application	application/json, application/x-www-form-urlencode
multipart	multipart/formed-data

2. 요청 몸체에 담아 서버로 전송할 페이로드의 MIME 타입을 지정하는 예

```
//XMLHttpRequest 객체 생성
const xhr = new XMLHttpRequest();

// HTTP 요청 초기화
xhr.open('POST', '/users');

// HTTP 요청 헤더 설정
// 클라이언트 서버로 전송할 데이터의 MIME 타입 지정: json
xhr.setRequestHeader('content-type', 'application/json');

// HTTP 요청 전송
xhr.send(JSON.stringfy({ id: 1, content: 'HTML', completed: false }));
```

43.3.4 HTTP 응답 처리

1. 서버가 전송한 응답을 처리하려면 XMLHttpRequest 객체가 발생시키는 이벤트를 캐치해야 함.

2. readyState 프로퍼티 값이 변경된 경우 발생하는 readystatechange 이벤트를 캐치하여 응답을 처리할 수 있다.

3. [예제 43-11]

```
//XMLHttpRequest 객체 생성
const xhr = new XMLHttpRequest();

// HTTP 요청 초기화
// Fake REST API를 제공하는 서비스
xhr.open('GET', 'https://jsonplaceholder.typicode.com/todos/1');

// HTTP 요청 전송
xhr.send();

//readystatechange 이벤트는 HTTP 요청의 현재 상태를 나타내는
//readyState 프로퍼티가 변경될 때마다 발생한다.
xhr.onreadystatechange = () => {
        //readyState 프로퍼티는 HTTP 요청의 현재 상태를 나타낸다.
        //readyState 프로퍼티 값이 4(XMLHttpRequest.Done)가 아니면
        //서버 응답이 완료되지 않은 상태다.
        //만약 서버 응답이 아직 완료되지 않았다면 아무런 처리가 발생하지 않는다.
  if(xhr.readyState !== XMLHttpRequest.Done) return;
```

```
  // status 프로퍼티는 응답 상태 코드를 나타낸다.
  // status 프로퍼티 값이 200이면 정상적으로 응답된 상태이고,
  // 200이 아니면 에러가 발생한 상태다.
  // 정상적으로 응답된 상태라면 response 프로퍼티에 서버의 응답 결과가 담겨 있다.
  if(xhr.status === 200) {
    console.log(JSON.parse(xhr.response));
    // { userId: 1, id: 1, title: "delectus aut autem", completed: fal
se }
  } else {
    console.error('Error', xhr.status, xhr.statusText );
  }
};
```

4. load 이벤트를 캐치하는 방법도 좋다.

```
// XMLHttpRequest 객체 생성
const xhr = new XMLHttpRequest();

// HTTP 요청 초기화
// https://jsonplaceholder.typicode.com fake REST API 제공 사이트
xhr.open('GET', 'https://jsonplaceholder.typicode.com');

// HTTP 요청
xhr.send();

// load 이벤트는 HTTP 요청이 성공적으로 완료된 경우 발생.
xhr.onload = () => {
    // status 프로퍼티는 응답 상태코드를 나타낸다.
    // status 프로퍼티 값이 200이면 정상적으로 응답된 상태
    // status 프로퍼티 값이 200이 아니면 에러가 발생한 상태
    // 정상적으로 응답된 상태라면 reponse 프로퍼티에 서버의 응답 결과가 담겨있다.
    if(xhr.status === 200) {
        console.log(JSON.parse(xhr.response));
    } else {
        console.error('Error', xhr.status, xhr.statusText);
    }
};
```